Contents

Introduction

Within this report I will explore the steps involved within the process of utilising a honeypot. This includes identifying what platform I will use to host my honeypot, the design and implementation of my honeypot, the analysis of the results and finally my recommendations and conclusion of how to set up a honeypot after going through the process.

Before gathering my log files and analysing the results, my honeypot will stay active and idle for about 30 days. This is to ensure that the data I receive is accurate and consistent to the fullest extent possible. The fact that no resource is produced or served by our honeypot means that all traffic is unauthorised and suspicious, and furthermore means that all data within the log files is interesting and useful (*Verma, 2003*).

Literature Review

Before I delve into my report, I will first describe what a honeypot is and how it works. A honeypot is a closely watched network decoy, which earns its value when being probed, attacked or potentially compromised (*Mokube & Adams, 2007*). There are several reasons why a honeypot would want to be utilised:

- It can be deployed along-side operational systems, it can divert attackers away from legitimate and sensitive systems.
- The logs can prove useful to understand similarities between attackers and their methods. In turn, this can be utilised to strengthen system security for currently operational systems.
- A honeypot can be used to recognize new attacks before they are used on real systems.

Generally there are two types of honeypots; research and production. Even though in my report I will only be utilising a research honeypot, I think it's important to understand both. A research honeypot is really in the name, it's purely for research purposes; it is there to collect methods and strategies used by the attacker. The honeypot is made up of fictitious data that appears sensitive and lucrative to hackers (*Production vs Research Honeypots: What's the Difference?*, 2020). On the other hand, a production honeypot aims to imitate real production services and resources, like a business network for example. As I have previously explained, these are commonly employed as a diversionary tactic, while also being used to allow a company to secure their real operational services further, with the vulnerabilities of the services pointed out within the honeypot (*Verma, 2003*).

In terms of a Honeypots value to an organisation, there are advantages and disadvantages as you'd expect. In terms of their limitations, a honeypot cannot be used as a prevention safeguard when it comes to defending a Cyber Attack, as the only preventative measure it utilises is possibly slowing down the attacker, and wasting their time (Zobal et al., 2019), and instead other systems should be used such as IDS (Intrusion and Detection System) or Firewalls. Another disadvantage of using a honeypot is that if it's not set up correctly, or that not the honeypot environment isn't realistic enough, an experienced hacker could recognise that he is not in a real system, and therefore avoid the honeypot (*What Is a Honeypot? Types, Benefits, Risks and Best Practices*, n.d.).

Design and Implementation

The honeypot I will be using for this project is the Linux-Based SSH honeypot called Cowrie. This is a low interaction, open-source honeypot which emulates a vulnerable SSH service. Cowrie is a very customisable honeypot allowing for it to imitate various versions of SSH. It logs all SSH activity, including but not limited to:

- login attempts with password and username combinations and all IP addresses associated in those attempts.
- Terminal Commands sent within the honeypot and downloaded files from the terminal.

SSH is a protocol used for Secure Remote Login/Access over possibly unsecure networks. It authenticates the user on the client-side and also includes multiple channels, meaning multiple connections can be made at the same time (*Ylonen, 2006*). SSH allows for a user to access resources remotely and also to be able to download resources using SCP (Secure Copy Protocol) which runs over SSH.

To allow my honeypot to be web facing, I set up a Linux Server on 'https://cloud.linode.com' (A Cloud Infrastructure Service Provider) as seen in **Figure 1.0**. The reason I set it up on a cloud server was to make sure there were no real resources connected to this honeypot, on the off chance that an attacker was able to break out of the environment and get access to sensitive information. It also allows the honeypot to be accessed constantly.

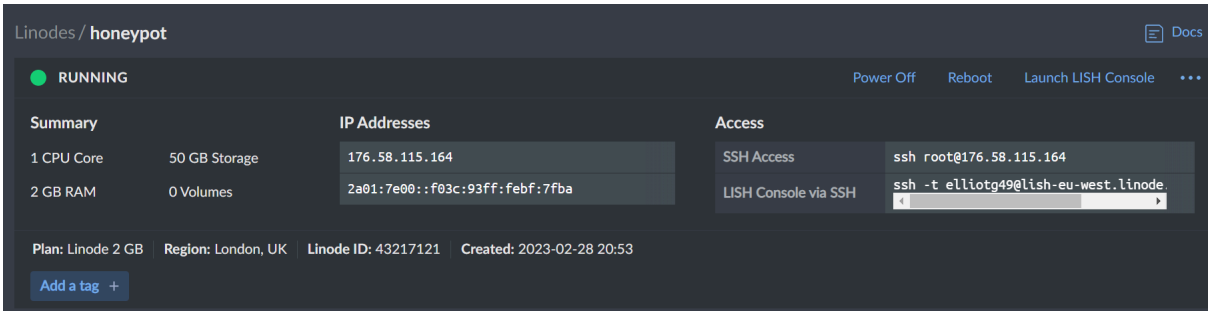


Figure 1.0 - Linode Web Interface showing a Honeypot Server

```
root@176-58-115-164: /etc/ssl

# This sshd was compiled with PATH=/usr/local/bin:/usr/bin:/bin:/usr/games

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 8888

#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

Figure 1.1 - sshd_config file showing default port

The first thing I did when I had access to my server was to change the default SSH port. This is so the honeypot can run on the standard SSH port (22) while I still have access to the server on a separate port (8888), so that log files can be accessed or the honeypot can be configured further. This can be understood in **Figure 1.1** and **Figure 1.2**

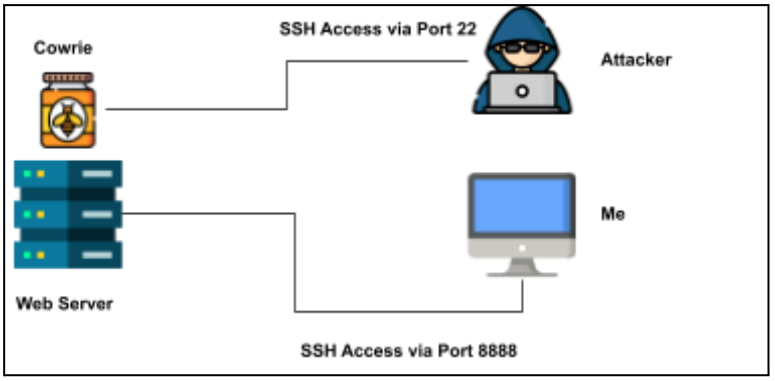


Figure 1.2 - Design Showing Access to Honeypot/Server through different ports

Once my server was active, I cloned the cowrie repository (https://github.com/cowrie/cowrie) onto the server. A few configuration changes had to be made to align cowrie with what I needed, for example the hostname of Cowrie is defaulted to ‘svr04’. This is a problem as because it's a default configuration, some attackers can recognise this and then realise they are in a honeypot environment. This defeats the purpose of this honeypot as we want the attacker to stay within the environment for as long as possible. Because of this I changed mine to ‘private ubuntu’ as seen in **Figure 1.3**.

```
# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment
#
# (default: svr04)
hostname = privateubuntu
```

Figure 1.3 - Hostname of Cowrie

As I previously mentioned, I changed my default SSH connection to port 8888, so I needed to change Cowrie's to 22, which was defaulted to 2222. From now on when I need to access the log files, I will have to specify the port within the SSH command 'SSH -p 888 root@'.

To allow Cowrie to listen on port 22, the service 'authbind' which allows non privileged root users (in this case the cowrie user) to bind network services (the honeypot virtual environment) to ports 1024 and under, had to be set up.

Cowrie produces its logs in two different formats '.log' and '.json'. Both of these have their Advantages and Disadvantages, but the main difference is that a .log produces a more readable format to the naked eye, and a json is less readable but includes more data, meaning it can be used for more complex data analysis.

Results and Analysis:

At the time of conducting this analysis, I have used 32 log files, therefore 32 days of data. The reason I started my honeypot this early in the process was to ensure that my results were as consistent and accurate as possible. Using this dataset, I have came up with four main aspects of which I will be investigating:

- Where these attackers are located geographically
- What these attackers do once inside my honeypot
- What type of attacks are happening in order to most effectively access my honeypot
- The most common password and usernames used

To retrieve useful information from all of these log files I used a number of python scripts, all of these scripts will be available on my github page which is located in this document. The aim of these results is to provide a broad overview of the data collected, to patterns and trends in the data, and to gain insight into the mind of an attacker.

The first aspect I investigated is the location of these attackers. This can be very useful information, as once you know the common locations you can identify potential threats a lot quicker, while also being able to respond to incidents much quicker, as a team can perform actions to prevent further damage.

I first used a python script to parse all IP addresses out of the logs (with duplicates being accounted for), and then used a linux based tool called ‘GeoIP’ to iterate through the list of IP addresses and geolocate each one. The reason I used this tool was due to many other services only allowing one IP address to be located at a time, and this would take a long time with 7000 IP addresses.

I’ve made 2 figures to demonstrate what countries attack my honeypot the most; **Figure 1.4** and **Figure 1.5**. My honeypot displayed a pretty low amount of diversity for this vector. With two countries displaying almost half of all attacks (China and the United States). But my results do concur with other conclusions that I've seen online with the US, China, Russia and South Korea being the top 4 leading countries. But it has to be known that IP Geolocators cannot determine 100% accuracy, and that even if an attack appears to be coming from one country, a VPN or Proxy could be being used and therefore skews our results.

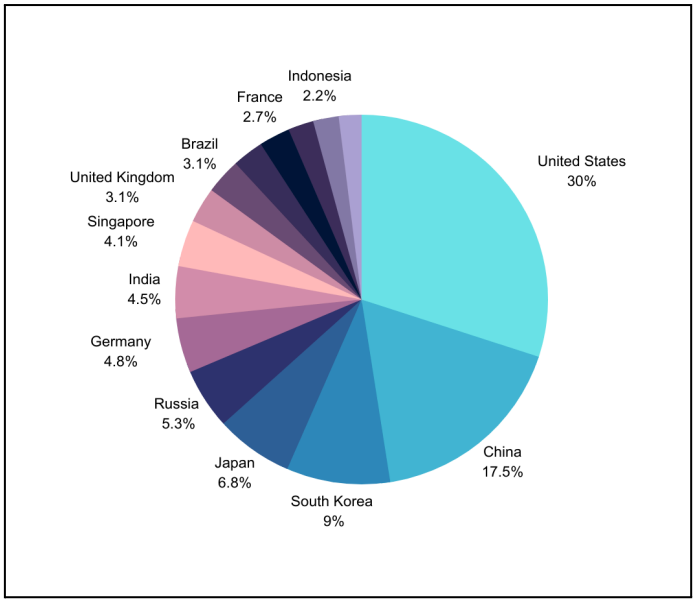


Figure 1.4 - Pie Chart showing most common countries

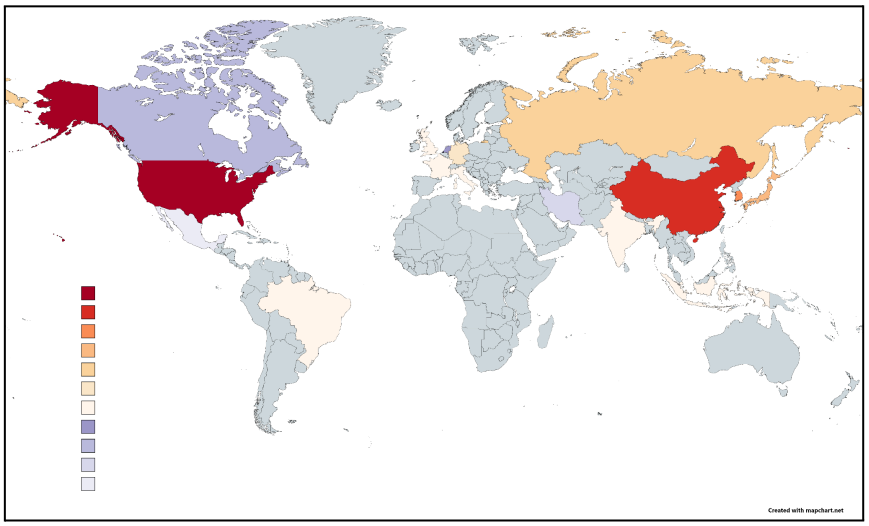


Figure 1.5 - Colour coded world map showing most common countries

Using these IP's I parsed from the json files, I also put them through Virus Totals (www.virustotal.com) IP reputation check, which checks if certain Security Vendors have flagged this IP previously. This can be useful as if you could automate this process, you could stop potential incidents before they happen. For example in **Figure 1.6** the IP address

‘107.170.234.14’ had been flagged by 7 previous vendors, meaning this IP is an ongoing security threat.

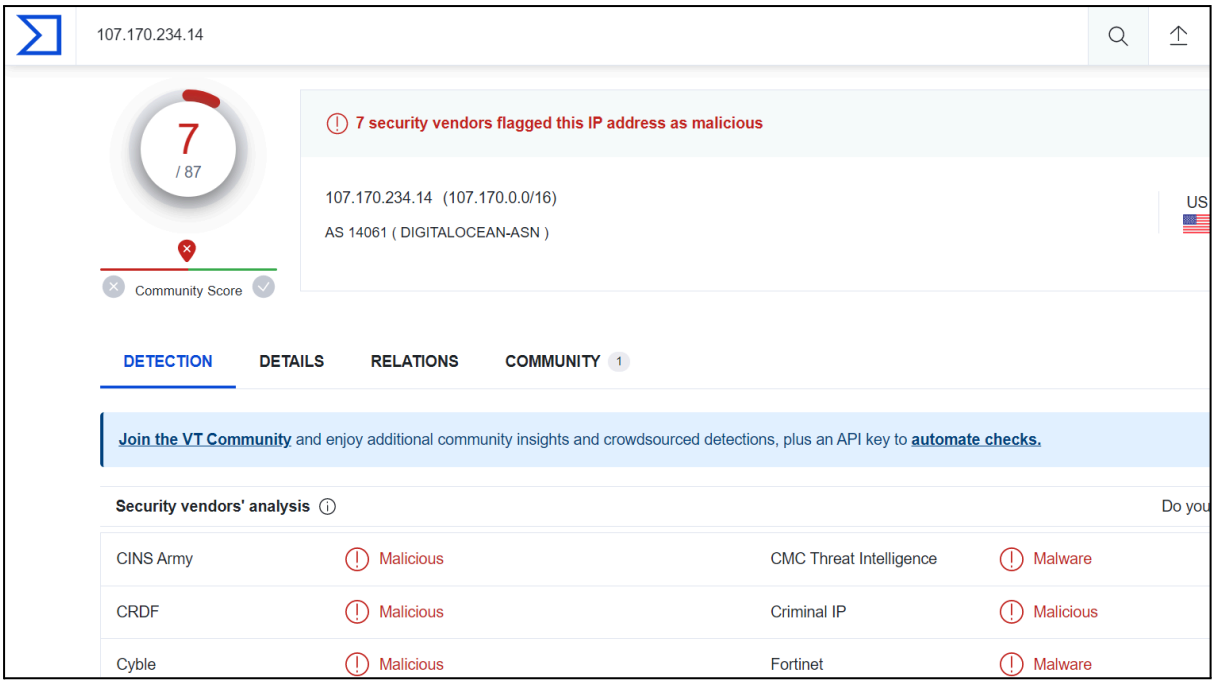


Figure 1.6 - VirusTotal Reputation Check

The second aspect I have investigated is what the attackers are doing once they are in my honeypot. With cowrie, not only does it save all commands used by the attackers, but any scripts or files which are downloaded within the environment are also saved in full. This means analysis can be done on said files. Researching and understanding these commands used by an attacker can provide useful intelligence for a network researcher or administrator, as you gain a valuable insight into the attackers mind due to seeing what the attackers overall end goal is.

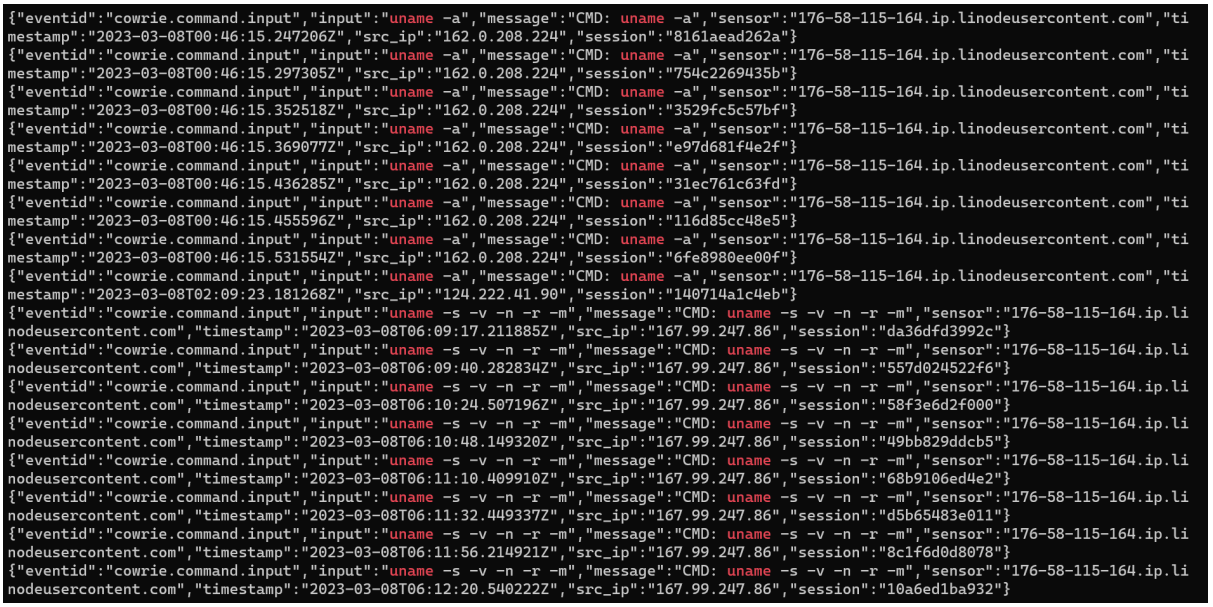
The most common done first by the attacker is adding their own SSH public key to the authorised_keys file within the ‘.ssh’ directory, as seen in **Figure 1.7.**

{"eventid":"cowrie.command.input","input":"cd ~ && rm -rf .ssh && mkdir .ssh && echo \"ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEArDp4cun2lhr4KUhBGE7VvAcwdli2a8dbnrTOrbMz1+5O73fcBOx8NVbUT0bUanUV9tJ2/9p7+vD0EpZ3Tz/+0kX34uAx1RV/75GVOmNx+9EuWOnvNoaJe0QXxziIg9eLBHpgLMuakb5+BgTFB+rkJAw9u9FSTDengvS8hX1kNFS4Mjux0hJOK8rvcEmPecjdySYMb66nylAKGwCEE6WEQHmd1mUPgHwGQ0hWCwsQk13yCGPK5w6hYp5zYkFnvlC8hGmd4Ww+u97k6pfTGTUbJk14ujvcD9iUKQTTWYYjIIu5PmUux5bsZ0R4WFwdIe6+i6rBLAsPKgAySVKPRK+oRw== mdrfckr\">>.ssh/authorized_keys && chmod -R go= ~/.ssh && cd ~","message":"CMD: cd ~ && rm -rf .ssh && mkdir .ssh && echo \"ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAQEArDp4cun2lhr4KUhBGE7VvAcwdli2a8dbnrTOrbMz1+5O73fcBOx8NVbUT0bUanUV9tJ2/9p7+vD0EpZ3Tz/+0kX34uAx1RV/75GVOmNx+9EuWOnvNoaJe0QXxziIg9eLBHpgLMuakb5+BgTFB+rkJAw9u9FSTDengvS8hX1kNFS4Mjux0hJOK8rvcEmPecjdySYMb66nylAKGwCEE6WEQHmd1mUPgHwGQ0hWCwsQk13yCGPK5w6hYp5zYkFnvlC8hGmd4Ww+u97k6pfTGTUbJk14ujvcD9iUKQTTWYYjIIu5PmUux5bsZ0R4WFwdIe6+i6rBLAsPKgAySVKPRK+oRw== mdrfckr\">>.ssh/authorized_keys && chmod -R go= ~/.ssh && cd ~","sensor":"176-58-115-164.ip.linodeusercontent.com","timestamp":"2023-03-08T14:29:38.955599Z","src_ip":"152.228.164.249","session":"1e1ee432a6e8"}

Figure 1.7 - Cowrie Command Event for adding Public Key

The reason an attacker would add their public key to my honeypot is to most likely preserve their presence, so if they need to access my honeypot again for further activity, they have automatic login. Also another reason for this is that if an attacker's initial way of entry gets

blocked or fixed, they still have access via public key. Another command/commands I found commonly used were reconnaissance type commands. For example ‘uname’, see **Figure 1.8** which produces the Operating System Version, hardware information, hostname and kernel info. The reason an attacker would do this is to further see if my system has any vulnerabilities, to maybe find multiple entry points, to again preserve their presence like before. Additionally another common command used was ‘history -c’, which clears the used commands log. This is not surprising as to preserve their access the attacker does not want the user to know that they have gained access.



{"eventid":"cowrie.command.input","input":"uname -a","message":"CMD: uname -a","sensor":"176-58-115-164.ip.linodeusercontent.com","ti
mestamp":"2023-03-08T00:46:15.247206Z","src_ip":"162.0.208.224","session":"8161aead262a"}
{"eventid":"cowrie.command.input","input":"uname -a","message":"CMD: uname -a","sensor":"176-58-115-164.ip.linodeusercontent.com","ti
mestamp":"2023-03-08T00:46:15.297305Z","src_ip":"162.0.208.224","session":"754c2269435b"}
{"eventid":"cowrie.command.input","input":"uname -a","message":"CMD: uname -a","sensor":"176-58-115-164.ip.linodeusercontent.com","ti
mestamp":"2023-03-08T00:46:15.352518Z","src_ip":"162.0.208.224","session":"3529fc5c57bf"}
{"eventid":"cowrie.command.input","input":"uname -a","message":"CMD: uname -a","sensor":"176-58-115-164.ip.linodeusercontent.com","ti
mestamp":"2023-03-08T00:46:15.369077Z","src_ip":"162.0.208.224","session":"e97d681f4e2f"}
{"eventid":"cowrie.command.input","input":"uname -a","message":"CMD: uname -a","sensor":"176-58-115-164.ip.linodeusercontent.com","ti
mestamp":"2023-03-08T00:46:15.436285Z","src_ip":"162.0.208.224","session":"31ec761c63fd"}
{"eventid":"cowrie.command.input","input":"uname -a","message":"CMD: uname -a","sensor":"176-58-115-164.ip.linodeusercontent.com","ti
mestamp":"2023-03-08T00:46:15.455596Z","src_ip":"162.0.208.224","session":"116d85cc48e5"}
{"eventid":"cowrie.command.input","input":"uname -a","message":"CMD: uname -a","sensor":"176-58-115-164.ip.linodeusercontent.com","ti
mestamp":"2023-03-08T00:46:15.531554Z","src_ip":"162.0.208.224","session":"6fe8980ee00f"}
{"eventid":"cowrie.command.input","input":"uname -a","message":"CMD: uname -a","sensor":"176-58-115-164.ip.linodeusercontent.com","ti
mestamp":"2023-03-08T02:09:23.181268Z","src_ip":"124.222.41.90","session":"140714a1c4eb"}
{"eventid":"cowrie.command.input","input":"uname -s -v -n -r -m","message":"CMD: uname -s -v -n -r -m","sensor":"176-58-115-164.ip.li
nodeusercontent.com","timestamp":"2023-03-08T06:09:17.211885Z","src_ip":"167.99.247.86","session":"da36dfd3992c"}
{"eventid":"cowrie.command.input","input":"uname -s -v -n -r -m","message":"CMD: uname -s -v -n -r -m","sensor":"176-58-115-164.ip.li
nodeusercontent.com","timestamp":"2023-03-08T06:09:40.282834Z","src_ip":"167.99.247.86","session":"557d024522f6"}
{"eventid":"cowrie.command.input","input":"uname -s -v -n -r -m","message":"CMD: uname -s -v -n -r -m","sensor":"176-58-115-164.ip.li
nodeusercontent.com","timestamp":"2023-03-08T06:10:24.507196Z","src_ip":"167.99.247.86","session":"58f3e6d2f000"}
{"eventid":"cowrie.command.input","input":"uname -s -v -n -r -m","message":"CMD: uname -s -v -n -r -m","sensor":"176-58-115-164.ip.li
nodeusercontent.com","timestamp":"2023-03-08T06:10:48.149320Z","src_ip":"167.99.247.86","session":"49bb829ddcb5"}
{"eventid":"cowrie.command.input","input":"uname -s -v -n -r -m","message":"CMD: uname -s -v -n -r -m","sensor":"176-58-115-164.ip.li
nodeusercontent.com","timestamp":"2023-03-08T06:11:10.409910Z","src_ip":"167.99.247.86","session":"68b9106ed4e2"}
{"eventid":"cowrie.command.input","input":"uname -s -v -n -r -m","message":"CMD: uname -s -v -n -r -m","sensor":"176-58-115-164.ip.li
nodeusercontent.com","timestamp":"2023-03-08T06:11:32.449337Z","src_ip":"167.99.247.86","session":"d5b65483e011"}
{"eventid":"cowrie.command.input","input":"uname -s -v -n -r -m","message":"CMD: uname -s -v -n -r -m","sensor":"176-58-115-164.ip.li
nodeusercontent.com","timestamp":"2023-03-08T06:11:56.214921Z","src_ip":"167.99.247.86","session":"8c1f6d0d8078"}
{"eventid":"cowrie.command.input","input":"uname -s -v -n -r -m","message":"CMD: uname -s -v -n -r -m","sensor":"176-58-115-164.ip.li
nodeusercontent.com","timestamp":"2023-03-08T06:12:20.540222Z","src_ip":"167.99.247.86","session":"10a6ed1ba932"}

Figure 1.8 - Cowrie Command Event for ‘Uname’ command

One frequent occurrence I noticed while analysing the log files was that attackers were commonly using commands such as ‘wget’ or ‘curl’ to download potentially malicious files from an external IP address or a Github Repository. I was interested in what type of malware the attackers would be accessing so I copied the downloaded malware from Cowrie into VirusTotals malware scanner. The results, shown in **Figure 1.9** indicated that this specific malware was a crypto miner (software used to mine cryptocurrencies). Out of the 10 files that were flagged as malicious, all of them were tagged as a trojan coin miner, meaning that the end goal of most of the attackers is to add me to a botnet (network of infected computers) of coin miners.

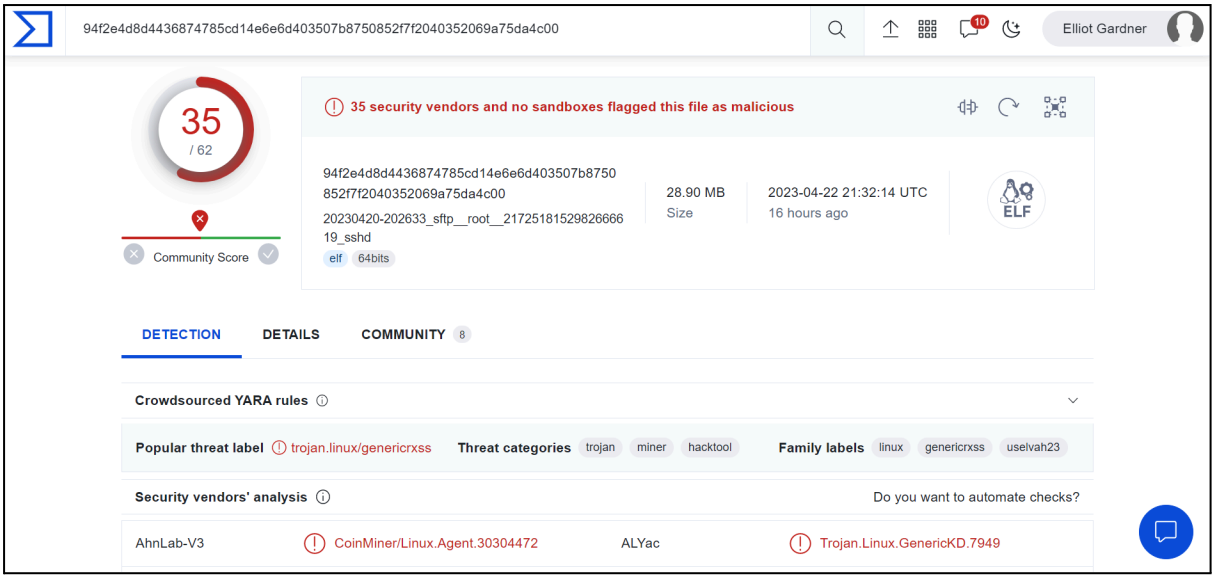


Figure 1.9 - VirusTotal showing malicious software results

As seen in **Figures 2.0 & 2.1** the most two most used username combinations were '345gs5662d34' and' admin', these are very commonly used username and password combinations in other systems, and therefore very likely that they are a part of a dictionary list for a dictionary type of attack. This is an attack using common and predictable usernames and passwords which are added to a large wordlist (Bosnjak et al., 2018). A way we could counteract this is a real system is by limiting the amount of login attempts each IP address is allowed, for example after 5 failed login attempts the IP address is blacklisted.

Username Entered	Username Count
345gs5662d34	11342
admin	3283
ubuntu	1577
user	1519
test	926
Pi	684
postgres	622
oracle	567
ubnt	555
ftpuser	473

Figure 2.0 - Table showing most common usernames entered

Password Entered	Password Count

345gs5662d34	11342
123456	9058
123	1702
1234	1636
password	1578
12345678	1163
123456789	996
12345	713
1	689
Password	663

Figure 2.1 - Table showing most common passwords entered

Knowing what software the attackers are using can be useful for multiple reasons, as attack patterns can be identified, also if many attacks come from a certain tool, and you know that anyone who is authorised to access the service you are providing doesn't use that tool , it can be blocked. As we can see ‘ssh-2.2-libssh_0’ is by the most used version of SSH that i encountered, it's very probable that this is because a vulnerability was found in this version (CVE-2018-10933) where attackers can bypass authentication without providing the proper credentials, and that the attackers are trying to exploit this vulnerability.

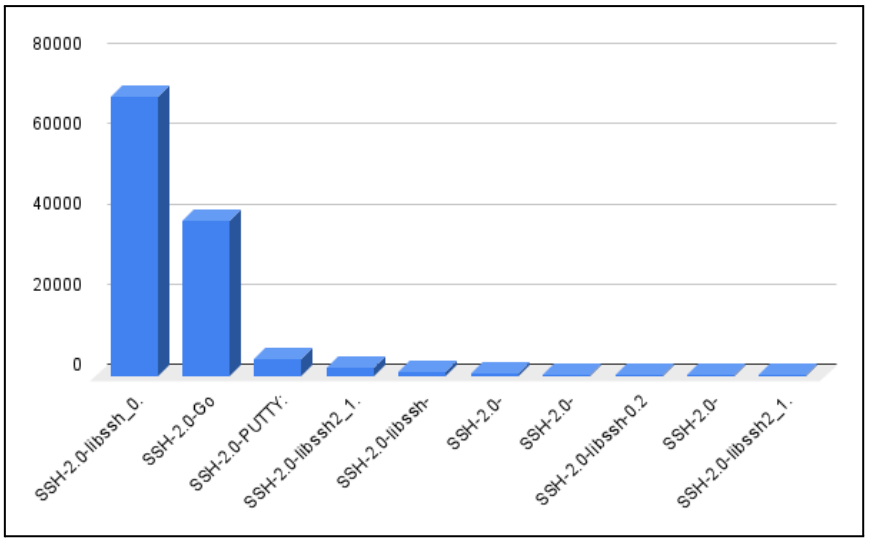


Figure 2.2 - Bar Chart Showing the most common versions of SSH used to access my honeypot

Recommendations

Many recommendations can be made due to the data I have collected from this honeypot. These are not recommendations for setting up a honeypot, but for setting up a legitimate system.

My first recommendation is to use an SSH key pair for all legitimate users, meaning that SSH auth by password can be completely disabled, and therefore stop 99% of attacks via SSH, since that is what my results show.

If this can't be fulfilled then secure passwords should be mandatory for all users, meaning that dictionary attack wordlists will not work for each user, especially the root/admin user.

Client and Server side restrictions should be put in place to stop/limit downloading malware from sources and deploying it. For example commands like ‘wget’ and ‘curl’ should be disabled for all non root users, or sufficient firewalls/anti viruses could be put in place to block the malware from being downloaded.

Lastly, if all legitimate users of a system are from similar geographical locations, Geographical Filtering can be used to block any login attempts from certain areas. For example a UK company could block all IP addresses from US, South Korea, China and Russia, like we saw from my honeypot results. Furthermore, if all legitimate users use an application like PuTTy to access to SSH server, then Version Filtering can be used to block any other requests

Conclusion

Based on the results found within my investigation, it’s clear there is an extreme level of attempted unauthorised access within public SSH systems. It's clear a honeypot which is being used for research purposes like in this occasion is incredibly useful for gaining intelligence into an attackers patterns, their insights and techniques they used. By analysing the data collected in the logs, it can be very useful in enhancing legitimate systems in order to mitigate the possibility of a Cyber Attack, as laid out in my recommendations. Lastly, because of the data I have laid out within this document, I do think all organisations should implement a honeypot into the Cyber Security Architecture, and should not be put off by being told a honeypot requires extensive knowledge to set up, as shown in my Design and Implementation section, little configurations had to be made.

References:

Bosnjak, L., Sres, J., & Brumen, B. (2018). Brute-force and dictionary attack on hashed real-world passwords. *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. https://doi.org/10.23919/mipro.2018.8400211

Mokube, I., & Adams, M. (2007). Honeypots. *Proceedings of the 45th Annual Southeast Regional Conference on - ACM-SE 45*. https://doi.org/10.1145/1233341.1233399

Production vs Research Honeypots: What's the Difference? (2020, June 22). Logix Consulting Managed IT Support Services Seattle. https://logixconsulting.com/2020/06/22/production-vs-research-honeypots-whats-the-difference/

Verma, A. (2003). *© S A N S I n s t i t u t e 2 0 0 4 , A u t h o r r e t a i n s f u l l r i g h t s . Security Essentials GSEC Practical Assignment Version 1.4b Option 1 Production Honeypots: An Organization's view Submitted by*. https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=69b4f3a6d1b8fa70962020b1959da9f0333c9f87

What is a Honeypot? Types, Benefits, Risks and Best Practices. (n.d.). Www.knowledgehut.com. https://www.knowledgehut.com/blog/security/honeypot

Ylonen, T. (2006). *The Secure Shell (SSH) Protocol Architecture*. https://doi.org/10.17487/rfc4251

Zobal, L., Kolář, D., & Fujdiak, R. (2019, October 1). *Current State of Honeypots and Deception Strategies in Cybersecurity*. IEEE Xplore. https://doi.org/10.1109/ICUMT48472.2019.8970921

Appendices

All code used to parse data out of log files can be located in my github repository here: https://github.com/elliog49/Honeypot-Project